Final Report on the Evaluation of StratusLab Products

Charles Loomis, Christophe Blanchet, Henar Muñoz Frutos, Evangelos Floros

Enhancing Grid Infrastructures with
Virtualization and Cloud Technologies

Final Report on the Evaluation of StratusLab Products

Deliverable D2.5 (V1.2)
14 June 2012

Abstract

Over the course of the project, the project's software releases have been evaluated against identified requirements and against the needs of real users. Most of the formal requirements have been satisfied, with work in the second year concentrating on extentions to commercial applications, multi-platform support and sandboxing. Feedback from applications running on the StratusLab cloud can be grouped into four broad categories: Ease of Use, Integration & Operation, Better Informaton Flow, and High-Level Services. These categories provide a broad roadmap for the evolution of the StratusLab cloud distribution past the end of the project.

StratusLab is co-funded by the European Community's Seventh Framework Programme (Capacities) Grant Agreement INFSO-RI-261552.

The information contained in this document represents the views of the copyright holders as of the date such views are published.

THE INFORMATION CONTAINED IN THIS DOCUMENT IS PROVIDED BY THE COPYRIGHT HOLDERS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE MEMBERS OF THE STRATUSLAB COLLABORATION, INCLUDING THE COPYRIGHT HOLDERS, OR THE EUROPEAN COMMISSION BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THE INFORMATION CONTAINED IN THIS DOCUMENT, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Contributors

Name	Partner	Sections
Charles Loomis	CNRS-LAL	All
Christophe Blanchet	CNRS-IBCP	3.1, 3.2
Henar Muñoz	TID	3.3
Vangelis Floros	GRNET	4.2

Document History

Version	Date	Comment
0.1	30 April 2012	Initial skeleton.
1.0	11 June 2012	Mostly complete version for review.
1.1	14 June 2012	Final draft after review.
1.2	14 June 2012	Minor corrections to bioinformatics sections.

Contents

List of Tables

1 Executive Summary

Both the scientific and commercial adoption of the StratusLab cloud distribution attest to the utility of its feature set and to the quality of the implementation. The broad range of scientific applications, including astrophysics, meterology, bioinformatics, and others, shows the broad applicability and interest in the project's software.

Using functional requirements collected at the beginning of the project as a yardstick, it is not surprising that the system has been well received. Most of the functional requirements were satisfied in the first year of the project with release v1.0. In addition to general service improvements, the second year concentrated on extentions to commercial applications, multi-platform support, and sandboxing. Moreover, identified performance bottlenecks related to network and disk bandwidth have been rectified.

The experiences from users and administrators have, however, identified areas in which further improvements and extensions can be made. At a macroscopic level, they can be grouped into four broad categories.

Ease of Use A recurring theme is simplifying the overall use of the cloud. The command line client has been well received by the community; however, they also desire a unified web interface. Requests to enhance the use-of-use have been made for other services as well, like enhanced search capabilities in the Marketplace.

Integration & Interoperation Cloud deployments occur in pre-existing environments and must adapt to the contraints of those environments, such as limited IP addresses and existing authentication mechanisms. One can also consider the need for standard OCCI and CDMI interfaces as a need to fit into a larger, federated cloud infrastructure.

Better Information Flow The exchange of information between the cloud provider and the cloud user needs to be improved, both at the operational level (e.g. notification of operational problems) and at the service level (e.g. providing comprehensive, synthesized accounting and billing statements).

High-Level Services Capturing usage patterns by offering high-level services makes the system more powerful. Areas where high-level services have been demonstrated are auto-scaling, federation, sandboxing, and multi-service deployment. These services can be further improved and expanded, for example by providing full multi-service workflows and orchestration.

As the project partners intend to continue working together on the StratusLab cloud distribution after the project's end, these four areas serve as a broad roadmap for the future evolution of the StratusLab software.

2 Introduction

Over the course of the project, the project's software releases have been evaluated against identified requirements and against the needs of real users. Basic tasks and workflows are documented in the project's tutorials and tested routinely as part of the build process, as well as regular certification of releases (see *Infrastructure Operations Final Report* (D5.5) [3]) This document concentrates on a more subjective evaluation based on deployed applications and on the requirements defined at the beginning of the project.

"Lessons learned" are extracted from a variety of applications deployed on the cloud infrastructure; these provide a set of refined requirements and recommendations that can be used to guide the further development of the StratusLab software after the end of the project. The applications themselves are described in the companion document *Final Report on StratusLab Adoption* (D2.4) [4].

The document also contains an evaluation of the software against the requirements collected at the beginning of the project. Many of the functional requirements were met with the v1.0 release. Work in the second year included improvements in stability and reliability, multi-platform support, and the scientific and commercial applications mentioned in the above document.

A complete description of the StratusLab cloud distribution itself can be found in the *StratusLab Toolkit 2.0* (D4.5) [2] document.

3 Use Cases

Seven use cases to evaluate the StratusLab service functionality and implementations had been previously defined. Five of the seven use cases have been implemented. These are described in the following sections along with the lessons learned from these implementations.

The unimplemented use cases concerned parallel and commercial computational chemistry applications on the cloud. Feedback on parallel applications and on the need for restricted applications (e.g. from a commercial license) have been collected from other scientific domains. Lessons learned from those applications and others are provided in the last section of this chapter.

To keep this document as concise as possible, the full descriptions of each use case have not been duplicated here. Please refer to the *Survey of Targeted Communities Concerning StratusLab* (D2.3) [1] for the full descriptions.

3.1 Bioinformatics Web Services

The adoption of clouds for bioinformatics applications will be strongly correlated to the capability of cloud infrastructures to provide ease-of-use and access to reference biological databases and common bioinformatics tools. This use case involves building customized appliances for this community and keeping them up-to-date. In addition, it is also necessary to integrate with the process (e.g. authentication) used with existing infrastructures to allow federation of these resources.

3.1.1 Lessons Learned

The deployment of these bioinformatics web services on the CNRS/IBCP infrastructure has highlighted missing functionality in the StratusLab cloud distribution. As these people are also directly involved in the project, they have also worked to provided solutions to fill these gaps.

- Port Address Translation: Many smaller sites have very few allocated public IP addresses, making the standard configuration of allocating each virtual machine a public IP address unworkable. Port Address Translation (PAT) for the ssh and http ports of virtual machines allows the machines to use local addresses but remain accessible from the outside. This has been developed and added to the distribution.

- Web Interface: Scientists want a simple web interface to the cloud that is

adapted to their needs. Such a portal was created that provides authentication mechanisms adapted to the community, specialized view of bioinformatics virtual machines, and control over cloud resources, showing that such customized interfaces can be created easily over the StratusLab services.

- Authentication: Each community has its own authentication mechanisms. For the bioinformatics infrastructure in France, Shibboleth can be used easily due to the national identity federation comprising all the universities and research institutes. Direct integration of Shibboleth in StratusLab turned out to be difficult. However, a deployment of a Short-Lived Credential Service (SLCS) from the grid world, allowed users to obtain short-lived certificates for use with the cloud from their Shibboleth credentials.

3.2 TOSCANI

TOSCANI: TOwards StruCtural AssignmeNt Improvement is a project to improve the determination of protein structures based on Nuclear Magnetic Resonance (NMR) information. The programs ARIA [6] and ISD [7, 8] are used to calculate the structures, which are computationally intensive. This application demonstrates the flexibility of the cloud to deploy the different bioinformatics tools required to accelerate such a procedure.

3.2.1 Lessons Learned

Different specific points have been identified with the TOSCANI use case. Some are about missing functionalities and others have highlighted important features that are already developed in StratusLab and that will ease the access of scientists to future e-infrastructures.

- Legacy Virtual Machines: Bioinformaticians may have their own, pre-built appliance that they want to use on the StratusLab infrastructure. If this appliance already uses KVM, then using it with StratusLab is quite simple; only the StratusLab contextualization tools and a reference in the Marketplace need to be added. However, pre-built appliances created with other virtualization systems are typically more difficult to integrate.

- Easy Platform Deployment: With the usual StratusLab interfaces (command-line client and web interface) users can launch multiple virtual machines that will be used as a platform for a particular bioinformatics analysis. The 'aria-clouder' script has been installed in the lastest release of the ARIA appliance to help users to configure ARIA parameter files to adapt to the created virtual infrastructure. This development helps the access of scientists to the cloud but should be made easier to allow such deployments from the web interface.

- Network Isolation: Having network isolation (dynamic VLANs) between the machine in a deployed platform, would simplify the creation and use of

the platform. For example, the platform may include an NFS file system, which is much easier to secure via network isolation than by configuration of the individual nodes.

- Pay as You Go: A structural biology laboratory not specially involved in bioinformatics likely would not invest in building a cluster ($\sim$100 nodes) to run NMR structure calculations occasionally. Thus, there is a strong interest in providing such tools, hosted at other institutes, to structural biologists on a 'pay as you go' basis. Adding comprehensive billing and accounting features to StratusLab framework would make the ARIA application more widely available and usable in the bioinformatics community. In France with the future French Institute for Bioinformatics, a mix of academic and commercial services is the foreseen model.

3.3 nTier Web Application Prototype

Modern internet applications are complex software implemented in several layers (tiers). In addition to the usual computing, storage, and networking resource expected in any IaaS cloud, they also require high-level services. These include: use of Key Performance Indicators (KPIs) to maintain a specific Quality of Service (QoS), auto-scalability, multi-tier management to manage the coupling between layers, and enhanced security.

3.3.1 Lessons Learned

The experimentation with this e-business application has provided a set of conclusions:

- Stateless Tiers: Scalability by the simple addition or removal of machines is only possible for tiers without state. By separating the data from the application itself, efficient scaling of applications can be more easily accomplished.

- Balancers for Scalability: Scaling a tier implies having a load balancer to control the number of replicas and to balance the load among them. Different balancing algorithms can be used and configuration of these balancers is required.

- Multi-tier Management: The application is deployed in different tiers each comprised of particular type of virtual machine. This complex structure including all of the tiers and the associated load balancers requires high-level deployment, configuration and management services. Claudia along with the StratusLab contextualization mechanism have allowed deployment of the different virtual machines and dynamic configuration of those virtual machines and load balancers. With just a click it is possible to have the whole service running.

- Persistent Application Data: Scaling down replicas means shutting down the associated virtual machines. To avoid data loss, data must be separated from the virtual machines in the tiers and stored persistently. In StratusLab, data are stored in the persistent disk services, so that, although virtual machines are deleted, the data are maintained.

- KPI-Based Scalability: The scalability focuses on service indicators; for instance, response time or number of users are good indicators about the service demand. Concretely in this use case, there have been scalability of each tier based on the same KPIs: number of simultaneous connections and the time to execute the request.

- Multi-Cloud Deployment: When using multiple cloud infrastructures, it is possible to request an "infinite" number of virtual machines. If the necessary resources are not available at one site, they can be found at another site (by federation or brokering), transparently for the user.

Current Cloud providers only offer VM deployment, but with StratusLab's advanced capabilities, as demonstrated by this multi-tier application use case, an organization can provision e-business applications (mainly composed by front-end, business-logic and database), automatically scale them based on service KPIs, ensure security and isolation, and maximize the available resources.

3.4 Software Engineering PaaS

Efficient software development requires a large number of supporting services– code versioning systems, bug trackers, and continuous integration servers, for example. StratusLab itself is an example for use of cloud services in software engineering. The initial scope was to see if StratusLab could become "self-supporting" to provide feedback to the project itself about the stability of its software and how easily they can be used to manage services.

3.4.1 Lessons Learned

The StratusLab's entire software engineering platform was not fully realized as a virtual platform running in the cloud, mainly to minimize disruption with the main development activities of the project. Nonetheless, tests were done with each part of the software engineering chain to see how easily cloud resources can be incorporated.

- Common Authentication: Having a consistent authentication system between all of the tools (in our case LDAP) is critical for creating a unified software engineering platform.

- Persistent Block Storage: Such a service makes it easy to isolate critical service state (e.g. service database) and to ensure it is persistent between virtual machine instances. Within the project, this was done with the LDAP

server, git repository, and JIRA. This would have been trival to use for other standalone services like Nexus, an artifact repository.

- Virtual Build Machines: Use of customized virtual appliances for build machines works very well, offering a wide variety of different platforms preconfigured with the required service dependencies. The multiplatform support for StratusLab was achieved this way.

- Autogenerated Build Appliances: The build machines were based on the base appliances generated by the project. It would have been easier if the build appliances were also generated automatically, saving the time to recreate the correct environment each time and having the procedure documented and stored.

- Build Appliance Instantiation: A StratusLab cloud plugin was created for Hudson (a continuous integration server) to automate the instantiation of build machine instances. The plugin worked, demonstrating auto-instantiation of build instances. However, this was never really used in production because the plugin API was fragile between Hudson releases, the Hudson control logic was too rudimentary, and the startup times too long (largely because prepared build instances were not available).

- Workflows Needed: Although Hudson allows for dependencies between jobs, this proved to be very fragile for the various build and test workflows required to create and to validate the StratusLab distribution. SlipStream probably would have helped somewhat here, but what is really needed is a data-based (in our case artifact and/or package based) workflow mechanism.

- Hypervisor Testing Bottleneck: Releases of KVM in mainstream operating systems do not allow KVM to be used within KVM[1]. Because of this physical nodes were required for testing the StratusLab software. This formed a bottleneck in the testing where all tests went through a single system. Moreover, this allowed us to test only a single cloud configuration.

In summary, individual software engineering services can easily be virtualized. Creating a platform requires common authentication as well as attention to details like providing build appliances to maintain agility of the entire system. The tools need to be improved to allow complete systems testing workflows to be maintained and run efficiently. Excepting rare cases where physical machines must be used, the cloud is an excellent basis for a software engineering platform.

3.5 EGI Integration

EGI defined a minimal set of six scenarios that provide a basis for use of virtualization and cloud technologies within the infrastructure and that promote an

[1] This is a restriction for nearly all full virtualizaton hypervisors.

incremental and evolutionary transition from the current infrastructure. These six scenarios are defined in the *EGI Cloud Integration Profile* document [5]. Table 3.1 lists these scenarios along with comments on the StratusLab implementation.

3.5.1 Lessons Learned

- IaaS Functionality Provided: The StratusLab functionality for instantiating virtual machines and interacting with storage provides the necessary basis for an Infrastructure-as-a-Service (IaaS) cloud.
- OCCI and CDMI in Default Deployments: Several OCCI implementations compatible with StratusLab are now available (OpenNebula, Venus-C, and EGI). They should be provided in the default StratusLab deployments and tested as part of the wider EGI testing effort to highlight any architectural and technical deficiencies.
- Extended Accounting: The accounting utilities should be extended to other StratusLab services (e.g. storage) to provide a comprehensive view of resource utilization. Messaging mechanisms should be provided to allow this information to be integrated into larger systems.
- "Billing": Users tend to forget about their virtual machines and allocated storage, leaving resources unused but also unavailable for others. Even simple reminders for users about their consumed resources would help enforce good behavior.
- Alarms: Often users were the first to spot problems on the StratusLab reference infrastructures. Proactive, periodic testing of the reference infrastructures, possibily with alarms, should be implemented. The existing use case tests serve as a basis for automating the necessary functionality tests.
- Expand Notification: The message-based notification prototype informs users when a machine changes state. This should be expanded to include email. A broader range of notification events, including for example unexpected changes of state and security alerts, should be implemented.

Overall the functionality corresponds well to the requirements for integrating cloud services in the EGI infrastructure. However, StratusLab should provide and/or improve the APIs, accounting information, and notification mechanisms to make tighter integration easier.

3.6 Feedback from Other Applications and Deployments

A companion document *Final Report on StratusLab Adoption* (D2.4) [4] provides descriptions of the major applications that have be used on the StratusLab reference infrastructures. It also provides a list of deployments of StratusLab-based

Table 3.1: *EGI Scenarios*

Running a pre-defined VM image	Functionality provided by StratusLab, although the chosen APIs (OCCI and CDMI) are not currently supported by StratusLab.
Running my VM image (with my data)	Provided by StratusLab through the Marketplace.
Deciding which virtualized resource to use	A brokering scenario is provided by Claudia, although the LDAP-based EGI registry is not currently supported.
Accounting across resource providers	Partially provided by OpenNebula accounting; accounting not available for other services. Collection of this information between providers is outside the scope of StratusLab services.
Reliability/availability of the resource	Defined use cases can be used as templates for Nagios tests to test the reliability and availability of services.
State change notification from the VM manager	Prototype implementation provided by StratusLab.

cloud infrastructures. Scientists, engineers, and system administrators have provided feedback, which has been synthesized into the points below.

3.6.1 Lessons Learned

- Licenced/Restricted Software: Many software packages have restrictions on their use. Because of this, the software cannot be included in the public appliance files without some protection. Initial investigations have shown that adding the software in an encrypted volume is one possible solution. The decryption keys are then controlled by the image creator and only given to those with the permission to run the software.

- Parallel Applications: Astrophysics and meterology applications have shown that the performance of parallel application on the cloud have significantly degraded performance. Some of this is due to the latencies and comparatively low bandwidth of 1-10 Gb/s networks compared to, for example InfiniBand. However, the performance should be studied systematically and use of paravirtualized drivers may help considerably.

- Propagating Errors: When problems occur with virtual machines in the cloud, it is often very difficult for a user to determine the cause. The client provides some error reporting, but often the errors are cryptic or non-existant. Improvements have been made as the software evolved, but this still remains a frustration for users.

- Operational Situation: At several points in time, there were operational problems with either the GRNET or LAL cloud infrastructures. Although a mailing list was created to inform people of these problems, this was not systematically used. A better mechanism of notifying cloud users needs to be devised.

- Improved Marketplace Searching: As the number of appliances in the Marketplace has grown, it has become more difficult to find an appropriate image. Further improvements should be made to the search features to allow users to more quickly find appliances of interest. The metadata should also be improved to encourage better descriptions of the appliances.

- Unified Web Interface: Although the command line client is simple to install and works well, many users would prefer a unified web-based interface. All of the StratusLab RESTful interfaces have a web interface. However these need to be integrated to provide a single endpoint for users.

- Documentation: The users have found that the documentation is still sparse, difficult to navigate, and often not up-to-date. More effort must be put into keeping the documentation organized and consistent with the current software releases.

- Roadmap: The Scrum development methodology was extremely productive for the project. But for those from the outside who are not following the process closely, it is extremely difficult to discern the overall roadmap. This is important as the software begins to be used more in production as system administrators need to plan upgrades and users need to understand when features will be available.

- Concentrate on Cloud Services: Although GRNET's production grid site running in the StratusLab cloud provides an excellent example of the cloud's potential, nearly all of the current deployments are to provide cloud services directly to users. Hence, the focus for future developments should be squarely in the cloud domain.

3.7 Summary

The variety of applications using the StratusLab cloud infrastructures has lead to a similarly rich set of recommendations or "Lessons Learned". These can be grouped into four broad categories.

Ease of Use A recurring theme is simplifying the overall use of the cloud. The python-based command line client has been well received by the community; however, they also desire a unified web interface to further simplify use of the system. Similarly, users have requested better capabilities for annotating appliance metadata descriptions and in parallel, easier and more targeted searching of the Marketplace database.

Integration and Interoperation Cloud deployments occur in pre-existing environments and must adapt to the contraints of those environments. Examples include the need for Port Address Translation (PAT) for sites with few public addresses and the need for Shibboleth (or other authentication mechanisms). One can also consider the need for standard OCCI and CDMI interfaces as a need to fit into a larger, distributed infrastructure like EGI.

Better Information Flow The exchange of information between the cloud provider and the cloud user needs to be improved. At the lowest level, providing more and better error feedback to the users would improve the user experience. Automating the information exchange through notification of cloud service problems and virtual machine state changes would further improve the administrator-user interactions. At the highest level, comprehensive, synthesized accounting and billing statements would provide a basis for discussions of resource provisioning, utilization, and allocation, potentially improving the platform for everyone.

High-Level Services Infrastructure-as-a-Service clouds provide very basic services that require a good understanding of system administration to use effectively. However, many usage patterns exist and providing high-level services corresponding to those patterns would make the system more powerful. Areas where high-level services have been demonstrated are with auto-scaling, federation, sandbox-

ing, and multi-service deployment. These services can be further improved and expanded, for example by providing full multi-service workflows and orchestration.

These four areas serve as a broad roadmap for the future evolution of the project's software. The detailed feedback provides a good basis for the improvement of individual StratusLab services.

4 Previously Identified Requirements

Near the beginning of the project, a survey of researchers and system administrators identified 25 requirements and recommendations. Tables 4.1–4.3 list these requirements concisely, indicate whether each is satisfied and provide detailed explainations. The full description of the requirements can be found in the document *Review of the Use of Cloud and Virtualization Technologies in Grid Infrastructures* [9].

4.1 Gaps

This document also identified four gaps that needed to be addressed in the second year. The work to fill each of these gaps is described below.

- *Performance and scalability: As the project has done with testing use cases, it also needs to develop an infrastructure to run performance and scalability metrics and to track these as the distribution evolves.*

  The project maintained the application benchmarks (which also contain performance benchmarks) in the second year. In addition, initial work was done to allow scalability testing on the Grid'5000 infrastructures. Unfortunately, other priorities such as identifying and supporting outside users left too little effort to implement systematic performance and scalability testing.

- *Storage: Services that provide storage must evolve into production-level services and expanded to include file-based access as well.*

  The prototype storage services in the v1.0 release were improved to fully production quality services in the second year. Moreover, these services were integrated into the image caching system to provide low-latency startup of virtual machines. They also provide the ability to customize and save images. File-based services were judged to be out-of-scope for the StratusLab distribution itself.

- *Network Services: These need to expand to provide better sandboxing of virtual machines, particularly via dynamic VLANs and dynamic firewalls.*

  These networking services were provided in the OpenNebula 3.2 release integrated with the StratusLab distribution. However, these services are not yet exposed through the StratusLab client and API.

- *Commercial users: More effort needs to be made to contact commercial users to ensure that their needs are met and to see if they are interested in adopting the StratusLab distribution.*

  SixSq has made significant progress in demonstrating the commercial possibilities of the StratusLab cloud distribution through its SlipStream product, the DS-Cloud Ready Pack (turnkey IaaS cloud), SCOS-2000 deployment with ESA contractors, and Atos Helix Nebula deployment. See the companion document [4] describing StratusLab adoption for details.

4.2 Performance Improvements

Although systematic performance monitoring was not implemented, specific performance problems were investigated and corrected.

For the network, one user noticed that the maximum bandwidth that he could achieve on the reference cloud was 100Mb/s. Initial measurements showed a slightly better bandwidth of 150 Mb/s, but still far short of the theoretical maximum for a 1 Gb/s Ethernet interface. Further investigation revealed that the cause of the problem is that the VM appliances are not configured with the virtio paravirtualized driver. Therefore, the network requests from the guests are emulated, which results in poor I/O performance. By activating virtio_net on both VM and hosting node the network bandwidth increases up to 680 Mb/s, more consistent with the interface's maximum.

Similarly, the access to the local disk passes through the native IDE driver and not through the paravirtualized virtio_blk driver. In this case, however, the performance differences are not significant.

As a result of this investigation, the project has ensured that all of the appliances created by the project contain the paravirtualized drivers. In addition, the client has been updated to allow users to specify the use of these drivers for better IO performance.

4.3 Summary

As can be seen from the tables, most of the functional requirements were satisfied in the first year of the project. The second year concentrated on extentions to commercial applications, multi-platform support, and sandboxing. In parallel, the service and client implementations were extended and improved.

Significant progress was made on all of the previously identified gaps. Two areas that need more work are the performance measures and integration of sandboxing features. The project did investigate and improve identified problems related to network and disk bandwidth. Advances in the sandboxing capabilities need to be fully integrated into the system and exposed through the client.

Although the remaining unimplemented and partially implemented requirements remain valid, the experience and feedback from real applications presented in the previous chapter and the previously identified gaps provide a better basis for

continued evolution of the StratusLab distribution after the project.

Table 4.1: *Requirements (I)*

Requirement	Year 1	Year 2	Comment
Effort to contact commercial enterprises		Full	SixSq has demonstrated commercial viability of StratusLab through its SlipStream product, the DS-Cloud Ready Pack (turnkey IaaS cloud), SCOS-2000 deployment with ESA contractors, and Atos Helix Nebula deployment.
Multiplatform support (RedHat & Debian)		Full	Three RedHat-based platforms are supported: CentOS, Fedora, and OpenSuSE. Support for Debian-based systems was determined not to be a priority.
Integration with site mgt. tools	Full		Integration with Quattor site management tools has been demonstrated (and used) since the first releases.
Demonstrate core grid services	Full		A production grid site over a StratusLab cloud has been maintained since early releases in the first year.
Base images for common OSes	Full	Extended	In the first year base images were provided for Ubuntu, CentOS, and ttylinux. Generation of these images was automated in the second year.
Cloud independent of VM OS	Full		Use of full virtualization allows complete independence of the guest and host operating systems.
CPU-intensive benchmarks	Full		Provided in application benchmark suite.
Workflow and master/worker benchmarks	Partial		The application benchmarks use the Kepler workflow system. A master/worker benchmark was not implemented.
IO oriented benchmarks	Full		Application benchmarks include standard tools for measuring disk and network bandwidth.
Access control	Partial		Access control based on user identity is provided by all services. Group and role access control is not exposed through StratusLab client or APIs.

Table 4.2: *Requirements (II)*

Requirement	Year 1	Year 2	Comment
File and block access to data	Partial		The persistent disk service provides block access to data. The project decided file/object storage was out of scope, as services from other cloud stacks (e.g. OpenStack Swift) could be used for this.
Access to object/relational databases			Specific services not implemented as part of StratusLab. Deployment of standard databases can be done with user appliances.
Year 1 concentrate on cloud development	Full		Most functional requirements were satisfied in Year 1 with improvements and extentions in the second year of the project.
Year 2 concentrate on cloud usage		Full	Most of the defined use cases [1] were implemented in year 2 with many additional example of StratusLab adoption [4].
Cloud service installation by end-users			Although the project has put significant effort in simplifying cloud installation by system administrators, the minimum resource requirements and required expertise means that this will still be difficult for end users.
Allow both full and para-virtualization	Partial		Full virtualization is support. In principle, para-virtualization is support, but there have been no requests for this and it is not tested by the project.
Command line interface and API	Full	Extended	A python command line interface is provided. All services except OpenNebula provide a RESTful API; OpenNebula exposes an XML-RPC interface. The command line interface and APIs were extended and refined in the second year.
Cloud must support broad range of services	Full		The project demonstrated production grid services running over the cloud in the first year.
Quantitative performance evaluations			Although the application benchmarks can be used for performance evaluations, the project did not use them systematically to evaluate the performance of services.
Mechanisms to trust machine images	Full	Extended	The Marketplace was provided in the first year of the project. This manages appliance metadata to facilitate trust between various actors. The site acceptance policy mechanisms were extended in the second year.

Table 4.3: *Requirements (III)*

Requirement	Year 1	Year 2	Comment
Consider all features in surveys	Full	Full	All of the requirements from the surveys were considered valid, although necessary prioritization means that some (as evident from this list) were not implemented.
Integration with site mgt. tools	Full	Extended	The integration with Quattor was demonstrated in the first year. The configuration was extended and modified to keep up with changes in cloud services implementations in the second year.
Cloud impl. must scale to O(10000) VMs			Scaling to this order has not been tested. A deployment on the Grid'5000 infrastructure was demonstrated as a prerequisite for scalability testing but was not completed because of other priorities.
Sandboxing of running VMs	Partial	Partial	The use of full virtualization provides a level of sandboxing on the physical machine (Y1). WP6 has developed services for dynamic firewalling and VLANs in the second year, but these are not yet visible through the StratusLab client.
Complete performance benchmarks	Partial		Application benchmarks also provide performance measures. Metrics based on these were not collected and analyzed regularly.
Use existing performance benchmarks	Partial		Application benchmarks use standard disk and network IO benchmarks. Metrics based on these were not collected and analyzed regularly.

Glossary

Appliance	Virtual machine containing preconfigured software or services
CDMI	Cloud Data Management Interface (from SNIA)
EGI	European Grid Infrastructure
Hybrid Cloud	Cloud infrastructure that federates resources between organizations
IaaS	Infrastructure as a Service
KPI	Key Performance Indicator
OCCI	Open Cloud Computing Interface
OVF	Open Virtualization Format
Public Cloud	Cloud infrastructure accessible to people outside of the provider's organization
Private Cloud	Cloud infrastructure accessible only to the provider's users
TCloud	Cloud API based on vCloud API from VMware
VM	Virtual Machine
XML-RPC	XML-based Remote Procedure Call

References

[1] M. Airaj, C. Blanchet, C. Loomis, T. Malliavin, H. Munoz, M. Nilges, and M. Sterzel. Survey of Targeted Communities Concerning StratusLab. http://stratuslab.eu/lib/exe/fetch.php/documents:stratuslab-d2.3-v1.0.pdf.

[2] M.-E. Bégin and K. Skaburskas. StratusLab Toolkit 2.0. http://stratuslab.eu/lib/exe/fetch.php/documents:stratuslab-d4.5-v1.0.pdf.

[3] E. Floros, S. Kenny, M. Airaj, G. Philippon, G. Tézier, R. Montero, and C. Loomis. Infrastructure operations final report. http://stratuslab.eu/lib/exe/fetch.php/documents:stratuslab-d5.5-v1.1.pdf.

[4] C. Loomis, M. Airaj, M.-E. Bégin, C. Blanchet, V. Floros, and C. Gauthey. Final Report on StratusLab Adoption. http://stratuslab.eu/lib/exe/fetch.php/documents:stratuslab-d2.4-v1.1.pdf.

[5] S. Newhouse and M. Drescher. EGI Cloud Integration Profile. https://www.egi.eu/indico/materialDisplay.py?materialId=1&confId=415.

[6] W. Rieping, M. Habeck, B. Bardiaux, A. Bernard, T. Malliavin, and M. Nilges. ARIA2: automated NOE assignment and data integration in NMR structure calculation. *Bioinformatics*, 23:381–382, 2007.

[7] W. Rieping, M. Habeck, and M. Nilges. Inferential structure determination. *Science*, 309:303–306, 2005.

[8] W. Rieping, M. Nilges, and M. Habeck. ISD: a software package for Bayesian NMR structure calculation. *Bioinformatics*, 24:1104–1105, 2008.

[9] StratusLab. Review of the Use of Cloud and Virtualization Technologies in Grid Infrastructures. http://stratuslab.eu/lib/exe/fetch.php/documents:stratuslab-d2.1-v1.2.pdf.